FINDING THE CONSTRAINED DELAUNAY TRIANGULATION
AND CONSTRAINED VORONOI DIAGRAM
OF A SIMPLE POLYGON IN LINEAR TIME

Francis Chin and Cao-an Wang

Technical Report TR-95-02

May 1995

DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF ENGINEERING
UNIVERSITY OF HONG KONG
POKFULAM ROAD
HONG KONG
This book was a gift from

Dept. of Computer Science
The University of Hong Kong
Finding the Constrained Delaunay Triangulation and Constrained Voronoi Diagram of a Simple Polygon in Linear Time

Francis Chin and Cao An Wang

Abstract

In this paper, we present a $\Theta(n)$ time worst-case deterministic algorithm for finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple $n$-sided polygon in the plane. Up to now, only an $O(n \log n)$ worst-case deterministic and an $O(n)$ expected time bound have been shown, leaving an $O(n)$ deterministic solution open to conjecture.

1 Introduction

Delaunay triangulation and Voronoi diagram, duals of one another, are two fundamental geometric constructs in computational geometry. These two geometric constructs for a set of points as well as their variations have been extensively studied [PrSh85, Aue91, BeEp92]. Among these variations, Lee and Lin [LeLi86] considered two problems related to constrained Delaunay triangulation $^4$: (1) the Delaunay triangulation of a set of points constrained by a set of non-crossing line segments and (2) the Delaunay triangulation of the vertices of a simple polygon constrained by its edges. They proposed an $O(n^2)$ algorithm for the first problem and an $O(n \log n)$ algorithm for the second one. While the $O(n^2)$ upper bound for the first problem was later improved to $\Theta(n \log n)$ by several researchers [Chew87, WaSc87, Seid88], the upper bound for second has remained unchanged and the quest for an improvement has become a recognized open problem [Aggr88, Aue91, BeEp92].

Recently, there have been some results related to this open problem on the Delaunay triangulation of simple polygons. Aggarwal, Guibas, Saxe, and Shor [AGSS89] showed that the constrained Delaunay triangulation of a convex polygon can be constructed in linear time. Chazelle [Chaz90] presented a

---

$^1$This work is supported by NSERC grant OPG0941629.
$^2$Department of Computer Science, The University of Hong Kong, Hong Kong.
$^3$Department of Computer Science, Memorial University of Newfoundland, St.John's, NFLD, Canada A1C 5S7.
$^4$Same as generalized Delaunay triangulation as defined in [LeLi86]
linear-time algorithm for finding an 'arbitrary' triangulation of a simple polygon. Klein and Lingas showed that this problem for \( L_1 \) metrics can be solved in linear time [KLi92], and this problem for the Euclidean metrics can be solved in expected linear time by a randomized algorithm [KLi93]. These efforts all seem to point toward a linear solution to the Delaunay triangulation of simple polygons and support the intuition that the simple polygon problem is easier than the non-crossing line segment problem.

In this paper, we settle this open problem by presenting a deterministic linear-time worst-case algorithm. Our approach follows that of [KLi93]: (i) first decomposing the given simple polygon into a set of simpler polygons, called pseudo-normal histograms, then (ii) constructing the constrained Delaunay triangulation of each normal histogram, and finally (iii) merging the constrained Delaunay triangulations of all these normal histograms to get the result. In this 3-step progress, the first and third were shown to be possible in linear time, but the second step was done in expected linear time by a randomized algorithm. Our contribution is to show how this second step can be done in linear worst-case time deterministically.

The organization of the paper is as follows. In Section 2, we review some definitions and known facts, which are related to our method. In Section 3, we concentrate on how to construct the constrained Delaunay triangulation or constrained Voronoi diagram of a normal histogram in linear time. We conclude the paper in Section 4.

2 Preliminaries

In this section, (i) we explain the constrained Delaunay triangulation problem and its dual, the constrained Voronoi diagram problem, (ii) we define pseudo-normal histograms, and (iii) to put our solution of how to construct the constrained Voronoi diagram of a pseudo-normal histogram into perspective, we explain the approach taken to first divide any simple polygon into pseudo-normal
histograms and then merge constrained Voronoi diagrams of these pseudo-diagrams for the solution of the original polygon.

2.1 Constrained Delaunay Triangulations and Constrained Voronoi Diagrams

The Constrained Delaunay Triangulation [LeLi86, Chew87, WaSc87, Seid88] of a set of non-crossing line segments $L$, denoted by $CDT(L)$, is a triangulation of the endpoints $S$ of $L$ satisfying the following two conditions: (a) the edge set of $CDT(L)$ contains $L$, and (b) when the line segments in $L$ are treated as obstacles, the interior of the circumcircle of any triangle of $CDT(L)$, say $\Delta ss's''$, does not contain any endpoint in $S$ visible to all vertices $s, s',$ and $s''$. Essentially, the constrained Delaunay triangulation problem is Delaunay triangulation with the further constraint that the triangulation must contain a set of designated line segments. Figure 1(a) gives the constrained Delaunay triangulation of two obstacle line segments and a point (a degenerated line segment). In particular, if $L$ forms a non-intersecting chain $C$, monotone w.r.t. a horizontal line $l$, we are only interested in the portion of $CDT(C)$ between $C$ and $l$. If $L$ forms a simple polygon $P$, only the portion of $CDT(P)$ internal to $P$ will be considered.

Given a set of line segments $L$, we can define the Voronoi diagram w.r.t. $L$ as a partition of the plane into cells, one for each of the endpoint set $S$ of $L$, such that a point $p$ belongs to the cell of an endpoint $v$ if and only if $v$ is the closest endpoint visible from $p$. Figure 1(b) illustrates the corresponding Voronoi diagram for the set of line segments given in Figure 1(a). Unfortunately, this Voronoi diagram is not the complete dual diagram of $CDT(L)$ [Aure91], i.e., some of the edges in $CDT(L)$ may not have a corresponding edge in this Voronoi diagram.

In [Seid88, Ling89, JoWa93], the proper dual for the constrained Delaunay triangulation problem has been defined as the Constrained (or called Bounded) Voronoi diagram of $L$, denoted by $V_c(L)$. It extends the standard Voronoi diagram by: (i) imagining two sheets or half planes attached to
each side of the obstacle line segments; (ii) for each sheet, there is a well-defined Voronoi diagram that is induced by only the endpoints on the other side of the sheet excluding the obstacle line segment attached to the sheet; (iii) the standard Voronoi diagram is augmented by the Voronoi diagrams induced by the sheets. Figure 1(c) gives an example of $V_e(L)$, the Voronoi diagrams on the plane and on the two sheets of the obstacle line segment $\overline{ab}$. Note that the Voronoi diagrams on the two sheets of the obstacle line segment $\overline{cd}$ happened to be the same as the Voronoi diagram on the plane. With this definition of $V_e(L)$, there is a one-to-one duality relationship between edges in $V_e(L)$ and edges in $CDT(L)$. It was further proved in [Seid88, JoWa93] that the dual diagrams, $CDT(L)$ and $V_e(L)$, can be constructed from each other in linear time. For simplicity, we omit the word 'constrained' over Voronoi diagrams in this paper as all the Voronoi diagrams are deemed to be constrained unless they
are explicitly stated to be standard Voronoi diagrams.

2.2 Pseudo Normal Histograms (PNH)

![Diagram](image)

(a) a normal histogram. (b) a pseudo-normal histogram.

Figure 2: Normal histogram and pseudo-normal histogram

A normal histogram (NH) [DjLi89] is a monotone polygon w.r.t. one of its edges, called bottom edge, such that all the vertices of the polygon lie on the same side of the line extending the bottom edge (Figure 2(a) gives an example). A pseudo-normal histogram (PNH) [KLi93] can be defined as a normal histogram with the first or last edge not monotone w.r.t. the bottom edge. Intuitively, a PNH can be viewed as a NH missing one of its bottom corners, i.e., a PNH can be transformed into an NH by adding a right-angle triangle at its bottom (Figure 2(b)).

2.3 Decomposition of a simple polygon into PNH's

Figure 3 illustrates how a polygon $P$ is decomposed into 13 PNH's. $PNH_1$ is associated with the vertical bottom edge $e$ missing its upper bottom corner; $PNH_2$, associated with the horizontal bottom edge $e'$, is missing its left bottom corner, etc.

A simple polygon $P$ with $n$ vertices can be decomposed into PNH's in $O(n)$ time according to [KLi93] when provided with what are known as the horizontal and vertical visibility maps of $P$ (Figure 4), which in turn can be obtained in linear time according to [Chaz90]. A diagonal of $P$ is a line segment joining two vertices of $P$ and lying entirely inside $P$, while a chord of $P$ is a line
segment: (i) lying entirely inside $P$, (ii) parallel to the designated bottom edge, and (iii) joining a vertex and a boundary point of $P$ (such a boundary point is called pseudo-vertex). The horizontal visibility map of a simple polygon $P$ is a set of chords trapezoidalizing $P$ so that every vertex of $P$ is associated with at most two chords. The vertical visibility map can be defined similarly.

The decomposition starts with an arbitrary edge $e$ of $P$ as the bottom edge of the first $PNH$. The interior of the $PNH$ refers to the part of $P$ illuminated by the parallel lights vertically to $e$ and emanating from $e \cup e_s$, where $e_s$ is one of the two edges (if any) incident to $e$ at an interior angle between $90^\circ$ and $180^\circ$. The boundary edges of the $PNH$ that are not edges of $P$ will be the bottom edges in the next step.

The decomposition of $P$ can then be represented by a tree such that each tree node is a $PNH$ and each tree edge represents the adjacency of two $PNH$'s sharing a chord. $PNH_1$, with an edge of $P$ as its bottom edge, is classified as the root. For each edge in $PNH_1$ which is not an edge of $P$, we regard it as the bottom edge for a son of $PNH_1$. $PNH_2$, $PNH_3$, and $PNH_4$ are sons of $PNH_1$ and whose bottom edges are all horizontal. Similarly, the grandsons of $PNH_1$ are those with vertical
bottom edges and adjacent to sons of $PNH_1$, etc.

2.4 Merging the Voronoi diagrams of $PNH$s

It has been proved [KIL93] that the Voronoi diagrams of every two $PNH$'s would not interfere with each other as long as these two $PNH$'s are (i) at the same depth not facing each other, or (ii) with their corresponding depths more than two apart. Since the sons of a $PNH$ on opposite sides facing each other can be separated by a horizontal (or vertical) line, the Voronoi diagram of the $PNH$ is first merged with the Voronoi diagrams of all its sons on one side and then with the Voronoi diagrams of all the remaining sons on the other. Condition (i) ensures that merging Voronoi diagram of the $PNH$ with all those of its sons in this way can be done in time linearly proportional to the total size of the $PNH$ and all its sons. Condition (ii) ensures that the Voronoi diagrams of adjacent $PNH$'s can be repeatedly merged together to obtain $V_c(P)$ in time linearly proportional to the size of $P$.

![Diagram](image)

Figure 4: Horizontal and vertical visibility maps of simple polygon $P$

In order to find $V_c(P)$ in deterministic linear time, what remains to be solved is the construction of the constrained Voronoi diagram of a pseudo-normal histogram efficiently. In [KIL93], a randomized algorithm is introduced to find the Voronoi diagram of an $NH$ in expected linear time. The Voronoi diagram of the corresponding $PNH$ can then be obtained by removing the bottom vertex from the Voronoi diagram of this $NH$ and this can be done in time linearly proportion to the size of the $NH$.

7
In the next section, we shall concentrate our effort to design a linear-time deterministic algorithm for constructing the Voronoi diagram of an NH.

3 Finding the Constrained Voronoi Diagram of an NH

Given a normal histogram \( H \) with a horizontal bottom edge \( e \), \( H \) is decomposed recursively into a tree, say \( T_i \), of smaller normal histograms called influence normal histograms (INH), where a node of \( T_i \) corresponds to an INH and an edge of \( T_i \) indicates an adjacency between two INH’s. In Figure 5, node 0 (the root INH) is \((v_1, v'_1, v_3, v'_3, v_5, v_6, v_7, v_8, v_9, v_{10}, v'_{12}, v_{12}, v_{13}, v'_{13}, v_{25}, v'_{25}, v_{26}, v'_{26}, v_{33}, v_{34}, v_3)\). Nodes 1-6 form the second level and are sons of node 0. Node 1 = \((v'_3, v_2, v_3)\), node 2 = \((v_3, v_4, v'_4)\), node 3 = \((v'_{12}, v_{11}, v_{12})\), node 4 = \((v_{13}, v_{14}, v'_4, v'_{13})\), node 5 = \((v_{25}, v_{26}, v_{27}, v'_{25})\), and node 6 = \((v_{28}, v'_{30}, v_{30}, v'_{22}, v_{32}, v'_{26})\). Nodes 7-9 form the third level with node 7 = \((v_{14}, v_{15}, v_{16}, v_{17}, v'_{17}, v_{19}, v_{20}, v_{21}, v_{22}, v_{23}, v_{24}, v'_{14})\), node 8 = \((v'_{31}, v_{29}, v_{30})\), node 9 = \((v'_{12}, v_{31}, v_{32})\). Node 10 = \((v_{17}, v_{18}, v'_{17})\) is on the fourth level.

The decomposition ensures that the portion of Voronoi diagram \( V_c(H) \) in each INH can only be affected by its own vertices and the vertices of its sons and nothing beyond. In general, the Voronoi cells of \( V_c(H) \) associated with vertices of an INH might cross its bottom edge and share edges with Voronoi cells associated with vertices of its parent, but not with those of its brothers nor its grandparents. Similarly, the Voronoi cells of an INH would not share any boundary with those of its grandsons. This property implies that, should the Voronoi diagrams of the INH’s \( (V_c(INH)) \) be given, the repeatedly merging of the Voronoi diagrams of the adjacent INH’s can be done in time linearly proportional to the sum of their sizes.

Let \( V(p) \) denote the Voronoi cell associated with vertex \( p \) in a Voronoi diagram. A point \( p \) in a normal histogram \( H \) is called an influence point if the Voronoi cell \( V(p) \) in \( V_c(H \cup \{p\}) \) will cross \( H \)’s bottom edge \( e \). The set of influence points is called the influence region \( IR \) w.r.t. bottom
edge e. Consider Figure 5, the IR of H w.r.t. $\overline{v_1v_5}$ (the bottom edge e) is the region enclosed by: $v_1v_5, v_5v_8, v_8v_7, v_7v_9, v_9v_9, v_{10}v_{25}, v_{35}v_{28}, v_{28}v_{34}, v_{34}v_{35}$, and $v_{35}v_1$, where $\overline{xy}$ and $\overline{yz}$ represent respectively the straight line and the arc joining vertices x and y. The root (or root INH) of $T_I$ is defined as the NH enclosing all influence points of H and containing only edges (or parts of edges) and chords of H such that all its horizontal chords would intersect the IR of H (i.e., the smallest NH containing IR). As an example, the root INH is indicated by the white region in Figure 5. Let us now consider the part of H excluding the root INH, which consists of zero or more disjoint polygons.

Each polygon is also an NH with a chord as its bottom edge. As given in Figure 5, H is decomposed into a root INH and 6 other NH's, i.e., the NH's above chords $\overline{v_1v_3}, \overline{v_3v_5}, \overline{v_1v_2}, \overline{v_1v_3}, \overline{v_{25}v_{28}}$, and $\overline{v_{28}v_{29}}$. For example, the NH above chord $\overline{v_{28}v_{29}}$ is $(v_{28}, v_{29}, v_{30}, v_{31}, v_{32}, v_{32})$. The decomposition can be recursively applied to each of these NH's.

Since any node of $T_I$ does not contain the influence points of its parent by the definition of INH, the Voronoi cell associated with a vertex of any node in $T_I$ could not cross the bottom edge of its parent. Thus, the part of $V_{\text{c}}(H)$ within the root can be formed by merging the Voronoi diagram of the root INH with those of its sons. As the Voronoi cells associated with the internal vertices of an INH never share any edges with the Voronoi cells of its brother INH (Theorem 1), the merging can be performed in $O(m_0 + \sum_{i=1}^{s} m_i)$ time, where $m_0$ is the number of vertices of the root, $s$ is the number of its sons and $m_i$ is the number of vertices of its ith son.

**Theorem 1** Let $v_1$ be a vertex of $\text{INH}_1$ with bottom edge $\overline{u_1u_7}$ and $v_2$ be a vertex of $\text{INH}_2$ with bottom edge $\overline{u_2u_2}$. Assume that $\text{INH}_1$ and $\text{INH}_2$ are brothers in $T_I$, $v_1 \neq u_1$, $v_1 \neq u_1$, $v_2 \neq v_2$, and $v_2 \neq u_2$. Then, the Voronoi cell of $v_1$ will never share any point with the Voronoi cell of $v_2$.

**Proof** By the property of normal histograms, without loss of generality, assume that $\overline{u_1u_7}$ is on the lefthand side of $\overline{u_2u_2}$, i.e., $u_1 < u_1 < u_2 < u_2$ according to their $x$-coordinates. We show that there does not exist a point $p$ in $H$ which is equidistant to $v_1$ and $v_2$ and no other vertex in $H$ is closer to $p$. 


than to $v_1$ and $v_2$. Since $p$ is in $H$, $p$ has to lie directly under $u_1w_1$ in order to be closer to $v_1$ than to $u_1$ or $w_1$, i.e., $u_1 < p < w_1$. Similarly, $p$ has to lie directly under $u_2w_2$, i.e., $u_2 < p < w_2$. Obviously, $p$ cannot simultaneously satisfy both conditions.

For example, the Voronoi cell of $v_{14}$ or $v'_{14}$ in $INH_4$ never share any point with the Voronoi cell of $v_{26}$ or $v_{27}$ in $INH_5$. Let $M(n)$ denote the merging time for constructing $V_c(H)$ with $|H| = n$ when provided with the Voronoi diagram of every $INH$ in $T_I$. Then, we have $M(n) = C \ast (m_0 + \sum_{i=1}^{s} m_i) + \sum_{i=1}^{s} M(n_i)$ where $C$ is a constant and $n_i$ is the number of vertices of the $i$th subtree. As $n = m_0 + \sum_{i=1}^{s} n_i$, we can show that $M(n) = C(2n - m_0)$ by induction. Thus, the total merging time is $O(n)$. Note that in the above calculation, the pseudo-vertices are also counted. Since $n$ is at most thrice the actual number of vertices of $H$ (as each vertex of $H$ might associate with at most two pseudo-vertices), the total merging time is still linearly proportional to the actual number of vertices of $H$.

![Diagram of INH's of H](image)

**Figure 5: Decomposition of H into INH's and T_I**
In the following sections, we shall prove the properties of influence region (IR) and influence normal histogram (INH) which allow us to do efficient merging and identification.

3.1 Influence Region.

Let \( H_V \) be a subpolygon of normal histogram \( H \), consisting of the bottom edge of \( H \) and all those vertices of \( H \) with the property that their associated Voronoi cells in \( V_c(H) \) cross the bottom edge of \( H \). \( H_V \) can be also viewed as the maximum subsequence of the vertices of \( H \) having this property. As \( H \) is an NH, \( H_V \) will also be an NH sharing the same bottom edge as \( H \). Let us consider the example given in Figure 5 again, \( H_V \) is indicated by the sequence of vertices \((v_1,v_5,v_6,v_7,v_8,v_9,v_{10},v_{25},v_{28},v_{34},v_{35})\).

Lemma 1 All points in \( H_V \) are influence points.

Proof By the definition of \( H_V \), the bisector of two adjacent vertices (except the two vertices of the bottom edge) of \( H_V \), which forms part of the \( V_c(H) \), always crosses the bottom edge of \( H \). In other words, \( H_V \) are partitioned by these bisectors into cells, each of which is associated with one of its vertices. These cells resemble the Voronoi cells of \( V_c(H) \). In fact, each of these cells in \( V_c(H_V) \) always includes its corresponding Voronoi cell in \( V_c(H) \). These bisectors also partition the bottom edge into segments according to their closest vertices in \( H \) or \( H_V \). It is sufficient to prove this lemma by showing that given any point \( z \) in \( H_V \), there always exists a point on the bottom edge which is closer to \( z \) than to any vertex in \( H \), i.e., \( V(x) \), the Voronoi cell of \( x \), in \( V_c(H \cup \{x\}) \) crosses the bottom edge.

Let \( z \) be a point in \( H_V \), in particular, in a Voronoi cell \( V(u) \), corresponding to vertex \( u \) in \( V_c(H_V) \). Furthermore, let the extended line of \( wz \) intersect the boundary of this Voronoi cell \( V(u) \) at \( y \), which may be a point on the bottom edge or a point on a bisector. If \( y \) is on the bottom edge, let \( z \) be \( y \), otherwise let \( z \) be the intersection point of that bisector and the bottom edge. As \( \angle uzz > 90^\circ \), \( z \) is closer to \( z \) than to \( u \). It is easy to see that \( z \) is closer to \( z \) than to any other vertices in \( H \), thus \( z \) is a
point on the bottom edge that belongs to $V(z)$, i.e., $V(z)$ crosses the bottom edge. Thus, $z$ belongs to $IR$.

In general, $IR$ includes some regions not belonging to $H_V$. Let $e$ be a boundary edge of $H_V$. If $e$ is also an edge of $H$, then $e$ must be an edge of $IR$, e.g., $\overline{v_5v_6}$, $\overline{v_6v_7}$, $\overline{v_7v_8}$, etc. in Figure 5. However, if $e$ is a diagonal of $H$, then $IR$ could include some region of $H$ above $e$. Each of these regions is defined as follows. Assume $e = \overline{uw}$ is a diagonal. $O_{\overline{uw}}$ denotes the region in $H$ above $e$ (i.e., outside $H_V$) and below the circular arc $\overline{uw}$ where its center is on the bottom edge. $O_{v_1v_2}$, $O_{v_9v_{25}}$, $O_{v_{25}v_{28}}$ and $O_{v_{28}v_{24}}$ are such examples in Figure 5.

**Theorem 2** $IR = (\cup_{\overline{uw}\in D} O_{\overline{uw}}) \cup H_V$, where $D$ is the set of edges of $H_V$ which are diagonals of $H$.

**Proof** By Lemma 1, we need to prove $IR - H_V = \cup_{\overline{uw}\in D} O_{\overline{uw}}$ (as $H_V \cap O_{\overline{uw}} = \phi$). Consider a point $p$ in $H$, but not in $H_V$. Then, point $p$ must lie above an edge $\overline{uw}$ of $H_V$ which is a diagonal of $H$.

On one hand, if point $p \in O_{\overline{uw}}$, then $b_{uw}$ and $b_{uw}$ will cross the bottom edge before intersecting each other, where $b_{xy}$ denotes the perpendicular bisector of vertices $x$ and $y$. Thus $p$ belongs to $IR$, i.e., $IR - H_V \supset \cup_{\overline{uw}\in D} O_{\overline{uw}}$. On the other hand, if $p \notin O_{\overline{uw}}$, then $b_{uw}$ and $b_{uw}$ will intersect each other above the bottom edge. Thus, $p$ does not belong to $IR$, i.e., $IR - H_V \subseteq \cup_{\overline{uw}\in D} O_{\overline{uw}}$. \[ \square \]

**Corollary** Given a normal histogram $H$ and let $H_V = (v_0, v_1, ..., v_n)$, influence region $IR$ w.r.t. $H$ can be defined by the same sequence of vertices of $H_V$ by replacing all diagonals $\overline{v_i v_{i+1}}$ of $H$ in the sequence of $H_V$ by an arc $v_i v_{i+1}$. \[ \square \]

### 3.2 Influence Normal Histogram (INH)

By definition, an $INH$ would contain all the edges of $IR$ or $H_V$ (Theorem 2) which are also edges of $H$ (e.g., $\overline{v_3v_5}$, $\overline{v_5v_7}$, $\overline{v_7v_8}$ etc. in Figure 5). As $O_{\overline{uw}}$ is part of $IR$ for every $\overline{uw} \in D$ (Theorem 2), the remaining edges of an $INH$ would be those chords and edges of $H$ enclosing $O_{\overline{uw}}$. Let $H_B$ be the
part of $INH$ above each diagonal $uv$ and enclosing $O_{uw}$. For example, as in Figure 5, the $H_B$'s are $(v_1, v'_2, v_3, v'_5, v_5), (v_{10}, v'_12, v_{12}, v_{13}, v'_13, v_{25}), (v_{25}, v'_28, v_{28})$ and $(v_{28}, v'_28, v_{33}, v_{34})$.

Now, we can have a precise description of $INH$. There are two types of vertices in $INH$, the vertices of $H_V$ and the vertices of $H_B$'s, with one $H_B$ for each edge in $D$. Thus, any vertex in $INH$ that is not in $H_V$ will be in $H_B$, and the endpoints of any edge in $D$ will be vertices in both $H_V$ and $H_B$. In the following we shall describe the properties of $H_B$ and $H_V$ and show that the Voronoi diagram of an $INH$ can be constructed in linear time.

A **monotonic histogram** is an $NH$ such that if the bottom edge is on the $z$-axis, then the $z$-coordinates of the vertices along the boundary are monotonically non-decreasing, and the $y$-coordinates of the vertices (except the last vertex) along the boundary are monotonically non-decreasing or non-increasing. A **bitonic histogram** is a composition of two monotone histograms such that the $z$-coordinates of the vertices along the boundary are monotonically non-decreasing, and the $y$-coordinates of the vertices along the boundary are first monotonically non-decreasing on one side and then monotonically non-increasing on the other.

**Lemma 2** $H_B$ is bitonic.

**Proof** Since $H_B$ is the smallest $NH$ enclosing $O_{uw}$, all its internal horizontal chords will intersect with $O_{uw}$, i.e., all its vertices, (except the top vertex and its associated pseudo-vertex/vertices), should be horizontally visible from $O_{uw}$. As $H_B$ consists of only edges (or parts of edges) and chords of $H$, all edges of $H_B$ should be monotonically non-decreasing in the $x$- and $y$-coordinates on one side and monotonically non-decreasing in the $z$-coordinate but monotonically non-increasing in the $y$-coordinate on the other. Thus, $H_B$ is bitonic. \(\square\)

**Lemma 3** The Voronoi diagrams of $H_B$ and $H_V$ can be constructed in linear time.

**Proof** It is shown in [DjLi89] that the Voronoi diagram of a monotonic histogram can be constructed
in linear time. By the fact that $H_B$ can be partitioned into two monotonic histograms by the vertical line through its highest vertex or edge (Lemma 2), the Voronoi diagrams of two such monotonic polygons can be merged in linear time [Wang93,KILi93]. Thus, $V_c(H_B)$ can be found in linear time. The Voronoi diagram of a $H_V$ can be constructed in linear time due to [AGSS89].

Note that in the construction of the Voronoi diagrams of $H_B$ and $H_V$, all the pseudo-vertices are ignored. Thus, the resulting Voronoi diagrams do not contain any Voronoi cell of pseudo-vertices. This approach is different from that proposed in [KILi93], which requires the removal of the Voronoi cells of pseudo-vertices.

The following lemma shows that the Voronoi diagrams of two $H_B$'s cannot affect each other.

**Lemma 4** Given an INH with its attached $H_B$'s, and let $x$ and $y$ be two vertices not belonging to $H_V$ but in two different $H_B$'s, then the Voronoi cells, $V(x)$ and $V(y)$, cannot share any point in $V_c(H_V)$.

**Proof** As $x$ and $y$ are vertices in two different $H_B$'s but not belonging to $H_V$, $x$ and $y$ must be separated by some vertex $z$ in $H_V$. By the definition of $H_V$, the Voronoi cell $V(x)$ must cross the bottom edge. Thus, $V(x)$ cannot share any point with $V(y)$ above the bottom edge. □

**Theorem 3** The Voronoi diagram of an INH can be constructed in time linearly proportional to its size.

**Proof** By Lemma 3, the Voronoi diagrams of $H_V$ and $H_B$'s can be constructed in time linearly proportional to their sizes. Since each $H_B$ shares an edge with $H_V$, the Voronoi diagrams of each $H_B$ and $H_V$ can be merged in time proportional to the number of Voronoi edges shared by them [Wang93,KILi93]. As different $H_B$'s do not interfere each other (Lemma 4), the total merging time is linearly proportional to the number of Voronoi edges shared by $H_B$'s and $H_V$, i.e., the size of INH. □
3.3 Region Identification

In this section, we shall present an algorithm which can identify the INH in an NH in time linearly proportional to the size of the INH. Chazelle's linear time algorithm [Chaz90] is first applied to the NH to obtain its horizontal visibility map (Figure 6). Because of the property of a normal histogram, H can be represented by a partition tree $T_P$, in which each tree node represents a chord in the map and each tree edge represents the adjacency of two chords. Let $n(v)$ denote the chord(s) associated with vertex $v$ of $H$. If there are two chords in $n(v)$, $n_L(v)$ and $n_R(v)$ denote the left chord and right chord respectively (Figure 6). With the partition tree, the INH to be identified can be represented as a rooted subtree of $T_P$ \(^5\). For example, the INH indicated by the shaded area can be represented by the rooted subtree as marked in Figure 6. The algorithm to identify the INH is based on tree traversal. In order to achieve linear time complexity, only those tree nodes relevant to the INH will be traversed. Thus, one of the key steps in the tree traversal is the pruning condition, i.e., when the traversal of a subtree can be terminated. The other key step is the identification of the vertices of $H_V$ so that we can partition the INH into $H_B$'s and $H_V$ for the construction of Voronoi diagrams as described in the previous section.

Let $l_v$ be the line segment from $v$ perpendicular to the bottom edge. The following two lemmas give sufficient conditions for a vertex $v$ of $H$ to be and not to be a vertex of $H_V$. Based on Theorem 2 and the definition of $H_B$, we can ensure that a visited vertex, that is not a vertex of $H_V$, shall be a vertex of $H_B$.

**Lemma 5** For any vertex $v$ of $H$, if $l_v$ does not intersect with any bisector $b_{uv}$, where $u \in (H - \{v\})$, then $v$ is a vertex of $H_V$.

**Proof** Line segment $l_v$ will lie entirely in the Voronoi cell associated with $v$, and the lemma follows directly from the definition of $H_V$. Vertices $v_b$ and $v_b$ in Figure 5 are such examples. \(\Box\)

\(^5\)A rooted subtree of $t$ has the property that the root of $t$ is also the root of the subtree.
Figure 6: An $INH$ and its tree $T_p$.

Note that even though $v$ does not satisfy the condition in Lemma 5, $v$ can also be a vertex of $H_V$ as long as we can show that the Voronoi cell $V(v)$ crosses the bottom edge.

**Lemma 6** Let $u, v, and w$ be vertices of $H$ in this ordering. If $v$ lies outside $O_{uw}$, or equivalently, bisectors $b_{uv}$ and $b_{uw}$ intersect each other above the bottom edge, then $v$ cannot be a vertex of $H_V$.

**Proof** Vertex $v$ lies outside $O_{uw}$ iff perpendicular bisectors $b_{uv}$ and $b_{uw}$ intersect each other above the bottom edge (case (2) of Figure 8). So the Voronoi cell of $v$ would not cross the bottom edge and the claim follows directly from the definition of $H_V$.

Vertex $v$ can lie outside $O_{uw}$ in two different ways, with $n(v)$ intersecting and not intersecting $O_{uw}$.

As $O_{uw} \subseteq IR$ (Theorem 2), if $n(v)$ intersects $O_{uw}$, $v$ will become one of the boundary vertices of $INH$.

Since $v$ is not in $H_V$, it must be a vertex of $H_H$. If $n(v)$ does not intersect $O_{uw}$, any vertex above $v$ cannot be a vertex of $INH$ as its associated chord will not intersect $O_{uw}$. Thus the tree traversal can be terminated at $v$. In fact, only the first (lowest) vertex during the tree traversal, whose chord does not intersect $O_{uw}$, can be a vertex of $INH$ or $H_B$. Based on this pruning condition, all the visited vertices will be either in $H_V$ or in $H_{B-\sigma}$.

Without loss of generality, assume the parent of $n(v)$ intersects $IR$ and $n(v)$ is being visited. Based
on Lemma 5 and Lemma 6, vertex \( v \) is tested and classified into one of the three types:

(a) a vertex in \( H_V \) (Lemma 5),

(b) a vertex in \( H_B \) (Lemma 6), or

(c) a potential vertex if we cannot decide whether it is in \( H_V \) or \( H_B \) yet.

Basically, if \( n(v) \) intersects \( IR \) then we shall identify \( v \) as a vertex of the \( INH \) and continue the tree traversal to visit \( v \)'s son(s). If \( v \) is the left(right)-endpoint of the chord, i.e., the \( IR \) is on its right (left)\(^6\), then the \( INH \) to be identified must be on the right(left)-hand side of \( v \) and \( v \) belongs to the left (right) boundary of the \( INH \). If \( v \) is a potential vertex, then \( v \) is put to the left (right) stack \( L_L(L_R) \) of vertices. As the chords of these two stacks of vertices always intersect \( IR \), vertices in \( L_L \) and \( L_R \) will eventually form the boundary of the \( INH \). These potential vertices in the stack will be determined later whether they belong to \( H_V \) or \( H_B \). In order to do so, the bottom element of the left (right) stack must be the latest left (right) vertex known to be in \( H_V \). Let us consider the example given in Figure 5, initially the left endpoint \( v_1 \) and the right endpoint \( v_{35} \) of the bottom edge of \( H \) form the bottom elements of stacks \( L_L \) and \( L_R \) respectively. After \( v_8 \) has been identified to be in \( H_V \), it can replace \( v_1 \) to be the bottom element of stack \( L_L \). Lemma 7 gives an important property of the potential vertices. Based on this property, the potential vertices are stored in the form of stacks, \( L_L \) and \( L_R \). The classification of \( v \) can be performed by only examining the two top elements, \( u_\ast \) and \( w_\ast \), of the left and right stacks, \( L_L \) and \( L_R \). Without loss of generality, let us consider only \( L_L \) in the following lemma.

**Lemma 7** Let \( L_L = (u_0, u_1, ..., u_k) \) be the stack of potential vertices, where \( u_0 \) is the bottom element of \( L_L \) and the only vertex of \( L_L \) in \( H_V \). The set of bisectors, \( b_{u_0u_1}, b_{u_1u_2}, ..., b_{u_{k-1}u_k}, b_{u_kw_\ast} \), would not intersect each other above the bottom edge, where \( w_\ast \) is the top element in \( L_R \).

**Proof** By induction on \( i \leq k \). Base step: when \( i = 1 \), bisectors \( b_{u_0u_1} \) and \( b_{u_1u_2} \) would not intersect

\(^6\)Note that if there are two chords associated with \( v \), \( v \) can be the right endpoint of one and the left endpoint of the other. The following discussion will still hold if we consider the chord one at a time.
each other above the bottom edge, otherwise \( u_1 \) would not be in \( L_L \) and would be in \( H_B \) (Lemma 6). Let \( u_{k+1} = w_\ast \) and assume that all bisectors \( b_{u_j u_{j+1}} \) for \( 1 \leq j \leq s - 1 \) and \( 1 < i \leq k \) do not intersect each other above the bottom edge, then they should not intersect \( b_{u_i u_{i+1}} \) either. Otherwise \( b_{u_i u_{i+1}} \) would have intersected \( b_{u_{i-1} u_i} \) above the bottom edge, vertex \( u_i \) should be in \( H_B \) (Lemma 6) and would not be in \( L_L \). Thus the lemma is proved.

When \( n(v) \) is visited, there are two actions to be taken: (Action A) Vertices of \( L_L \) and \( L_R \) are tested one by one against \( v \) to determine whether any of these vertices belong to \( H_B \), and (Action B) vertex \( v \) is then tested against \( L_L \) and \( L_R \) in order to classify whether or not \( v \) is a vertex in \( H_V \), a vertex in \( H_B \), or a potential vertex. With the property described in Lemma 7, we shall show that the above tests can be carried out on the two top elements, \( u_\ast \) and \( w_\ast \), of \( L_L \) and \( L_R \) respectively, instead of all vertices in \( L_L \) and \( L_R \) or all vertices in \( H \) as stated in Lemma 5 and Lemma 6. Before describing the tests in detail, we shall consider an example as given in Figure 7 to illustrate the possible scenarios for how \( v \) is tested against \( L_L \) and \( L_R \). Starting with \( u_0 \), a vertex in \( H_V \), as the bottom element in \( L_L \), and after visiting \( n(u_1) \), \( n(u_2) \) and \( n(u_3) \), we still cannot decide whether or not \( u_1, u_2 \) and \( u_3 \) are vertices of \( H_V \) or \( H_B \). Thus, vertices \( u_1, u_2 \) and \( u_3 \) are potential vertices and pushed to the stack \( L_L \). When \( v \) is visited, four different scenarios shown in Figure 7 are possible:

![Figure 7: Some possible scenarios when \( v \) is visited](image-url)
(a) \( u_1, u_2, \) and \( u_3 \) are all popped from \( L_L \) as they are vertices in \( H_B \) (Action A), and vertex \( v \) as a potential vertex will be pushed to \( L_L \) (Action B),

(b) all \( u_1, u_2, \) and \( u_3 \) remain in \( L_L \) as potential vertices (Action A). We can show later that \( v \) belongs to \( H_V \) by Lemma 5 when \( u_* \) is also considered (Action B),

(c) \( u_1 \) and \( u_2 \) remain in \( L_L \) as potential vertices, and vertex \( u_3 \) is popped from \( L_L \) as it is in \( H_B \) by Lemma 6 (Action A). We can show later that \( v \) belongs to \( H_V \) by Lemma 5 when \( u_* \) is considered (Action B),

(d) same as (b) except that we can show that \( v \) is in \( H_B \) by Lemma 6 (Action B), and the tree traversal can be terminated at \( v \).

Let \( L_L = (u_0, u_1, \ldots, u_k) \) be the left stack of potential vertices just before \( v \) is visited. Action A is to remove from \( L_L \) those vertices identified to be in \( H_B \). Vertex \( u_k \) is first tested against \( v \) to see whether \( u_k \) is in \( H_B \) or remains in \( L_L \). Vertex \( u_k \) is in \( H_B \) if \( b_{u_k-1,u_k} \) and \( b_{u_k,v} \) intersect each other above the bottom edge (Lemma 6). If \( u_k \) is found to be in \( H_B \), the test will be carried on with \( u_{k-1} \) and so on until we find the first \( u_{k'} \), \( 0 \leq k' \leq k \), that remains in \( L_L \), i.e., \( b_{u_{k'-1},u_{k'}} \) does not intersect with \( b_{u_{k'},v} \) above the bottom edge.

**Action A:** while \((k \geq 1) \) and \( (b_{u_{k-1},u_k} \) intersects \( b_{u_{k},v} \) above the bottom edge)) do

begin pop \( u_k \) off \( L_L \) as \( u_k \) is a vertex of \( H_B \); \( k \leftarrow k - 1 \) end

Lemma 7 guarantees that if \( b_{u_{k-1},u_k} \) does not intersect \( b_{u_{k},v} \) above the bottom edge, \( b_{u_{p-1},u_p} \) would not intersect with \( b_{u_{p},v} \) above the bottom edge for all \( p, \ 0 \leq p \leq k \). Thus the test can be stopped at the first element in \( L_L \) that cannot be identified as a vertex in \( H_B \).

Note that some vertices in \( L_L \) might be known to be in \( H_V \), but for the simplicity of the algorithm, they remain in \( L_L \) as potential vertices and be recognized as vertices in \( H_V \) later.

After applying Action A to \( L_L \), vertices in \( L_L \) should possess a stronger property than that stated
in Lemma 7. (Note that the value of \( k \) may be smaller after Action A.)

**Lemma 8** After the application of Action A on \( L_L \), assume \( L_L = \{u_0, \ldots, u_k\} \) when \( v \) is visited. Then, the set of bisectors, \( b_{u_0u_1}, b_{u_1u_2}, \ldots, b_{u_{k-1}u_k}, b_{u_k1} \), would not intersect each other above the bottom edge.

The following lemma and theorem show that we can classify \( v \) as a vertex in \( H_V \), in \( H_B \), or a potential vertex by considering the top elements of \( L_L \) and \( L_R \) (Action B).

**Lemma 9** After the application of Action A on \( L_L \), let \( u_\ast \) and \( w_\ast \) be the top elements in \( L_L \) and \( L_R \) respectively when \( v \) is visited,

(a) \( b_{u_\ast v} \) does not intersect with \( l_v \) iff \( b_{w_\ast v} \) does not intersect with \( l_v \) for all \( z \in L_L \),

(b) \( b_{w_\ast w} \) does not intersect with \( l_v \) iff \( b_{z_\ast v} \) does not intersect with \( l_v \) for all \( z \in L_R \), and

(c) both \( b_{u_\ast v} \) and \( b_{w_\ast w} \) do not intersect with \( l_v \) iff \( b_{z_\ast v} \) does not intersect with \( l_v \) for all \( z \in H \).

**Proof** (a) "If" part is straightforward. "Only if" part: Let \( L_L = (u_0, u_1, \ldots, u_k) \) with \( u_\ast = u_k \). The proof is by induction on \( i \) that \( b_{u_k-1v} \) would not intersect with \( l_v \). It is true for \( i = 0 \) as \( b_{u_kv} \) does not intersect with \( l_v \). By Lemma 7, the set of bisectors \( b_{u_0u_1}, b_{u_1u_2}, \ldots, b_{u_{k-1}u_k} \) would not intersect each other above the bottom edge. Assume the hypothesis is true for some \( i > 0 \), i.e., bisector \( b_{u_{k-1}} \) does not intersect with \( l_v \). As bisector \( b_{u_k-(i+1)v} \) must lie between bisectors \( b_{u_k-(i+1)u_{k-1}} \) and \( b_{u_{k-1}v} \), bisector \( b_{u_k-(i+1)v} \) cannot intersect \( l_v \). Thus, (a) is true for \( i + 1 \).

(b) Similar to (a).

(c) For the visited vertices \( z \) visible from \( v \) and identified to be in \( H_V \), bisector \( b_{zv} \) cannot intersect \( l_v \) by a similar argument as given in (a).

As for those vertices \( z \) visible from \( v \) and identified to be in \( H_B \), if \( z \) lies between two vertices \( u_{i-1} \) and \( u_i \) in \( L_L \) or \( L_R \), then bisector \( b_{zv} \) must lie between bisectors \( b_{u_{i-1}v} \) and \( b_{u_iu} \). From (a), we have that bisector \( b_{u_\ast v} \) does not intersect with \( l_v \), \( b_{zv} \) cannot intersect \( l_v \). The proof is similar if \( z \) lies between two vertices in \( H_V \).
For those unvisited vertices $z$ above $v$, i.e., whose chords are in the subtree of $T_P$ rooted at $n(v)$, and visible from $v$, bisector $b_{zv}$ will slope further away from $l_v$ and hence cannot intersect with $l_v$.

For those vertices $x$ invisible from $v$, there must exist a vertex $y$ which satisfies the following properties: (i) $y$ lies between $x$ and $v$ when traced along the boundary of $P$, (ii) $y$ belongs to one of the above mentioned types of vertices with the property that bisector $b_{yv}$ does not intersect with $l_v$, and (iii) $y$ is visible from $v$. As $b_{zv}$ will lie further away from $l_v$ than $b_{yv}$, $b_{zv}$ cannot intersect $l_v$. □

**Theorem 4** After the application of Action $A$ on $L_L$ when $v$ is visited, let $u_*$ and $w_*$ be the top elements in $L_L$ and $L_R$ respectively. Then, one of the following cases can happen (note that $n^L(v)$ and $n^R(v)$ might be null)

(a) if none of $b_{uv}$ and $b_{uw}$ intersect with $l_v$, then $v \in H_V$,

(b) if both $b_{uv}$ and $b_{uw}$ intersect each other above bottom edge, then $vcH_B$, and

(c) if neither (a) nor (b) is satisfied, then $v$ is a potential vertex.

**Proof** (a) By Lemma 9(c), $b_{xv}$ would not intersect with $l_v$ for all $x \in H$ and $v$ would be a vertex in $H_V$ (Lemma 5). (b) The intersection of $b_{uv}$ and $b_{uw}$ above the bottom edge implies that $v$ would lie outside $O_{uw}$. Since it is assumed that the parent of $n(v)$ intersects $IR$, either $n(v)$ intersects $IR$ or $n(v)$ is the first (lowest) chord during the tree traversal not intersecting $IR$. Thus $vcH_B$ by the definition of $INH$. (c) By the definition of potential vertex. □

Before describing the whole algorithm in detail, we shall consider how and why the tree traversal works. The pruning only applies to those chords above edges in $D$, i.e., edges in $H_V$ and diagonals of $H$ and that do not go through the $IR$. For example, as shown in Figure 5, $n(v_3)$ above diagonal $\overline{v_1v_8}$, $n(v_{12})$ and $n(v_{13})$ above diagonal $\overline{v_{10}v_{25}}$, $n(v_{25})$ above diagonal $\overline{v_{25}v_{28}}$ and $n(v_{28})$ above diagonal $\overline{v_{28}v_{34}}$. By the time the nodes in $T_P$ corresponding to those chords are visited, the endpoints of their associated diagonals, i.e., edges in $D$, should have been traversed and identified. For example, when $n(v_{12})$ and $n(v_{13})$ are visited, $n(v_{10})$ and $n(v_{25})$ should have been visited. Thus we can determine the
IR extended above the diagonal $\frac{v}{25}$, i.e., $O_{10\sqrt{25}}$, and whether a chord above that diagonal can be pruned or not. When $v$ is visited, and after the application of Action A on $L_L$ and $L_R$, the bisectors $b_{uv}$ and $b_{vw}$, are closely related to the pruning. If $b_{uv}$ intersects $l_v$ above the bottom edge, $O_{uv}$ would lie totally below $n^L(v)$ and pruning can be applied at $n^L(v)$. Alternatively, if $b_{uv}$ intersects the bottom edge and not $l_v$, some portion of $O_{uv}$ would lie above $n^L(v)$ and pruning cannot be applied at $n^L(v)$. Similarly for $b_{uw}$ and $l_v$.

**Theorem 5** After the application of Action A, let $u_*$ and $w_*$ be the top elements in $L_L$ and $L_R$ respectively. We have the following cases will happen when $v$ is being visited (note that $n^L(v)$ and $n^R(v)$ might be null) (refer to Figure 8)

(a) The subtree rooted at $n^L(v)$ is pruned during tree traversal iff $b_{uv}$ crosses $l_v$.

(b) The subtree rooted at $n^R(v)$ is pruned during tree traversal iff $b_{vw}$ crosses $l_v$.

![Figure 8: The cases on pruning](image)

**Proof** The proofs for (a) and (b) are similar and only the proof for (a) is shown. Bisector $b_{uv}$ intersects with $l_v$ iff $O_{uv}$ lies totally below $v$ iff all chords above $n^L(v)$ do not intersect $IR$ iff $n^L(v)$ can be pruned. \[\square\]

The above theorem describes the condition when a subtree can be pruned. Subtrees are pruned as long as their corresponding $INH$'s do not contain any $IR$ region. When a subtree is pruned, the corresponding pruned portion of the normal histogram is a smaller normal histogram with the pruned chord, i.e., the root of the subtree, as its bottom edge. For example, the pruned portion
\( (v_{28}, v_{29}, v_{30}, v_{31}, v_{32}, v_{28}) \) is an \( NH \) with \( \overline{v_{28}v_{28}} \) (the pruned chord) as bottom edge. The Voronoi diagrams of pruned portions can then be constructed recursively.

We shall describe Action B to be taken when \( v \) is visited after the application of Action A on \( L_L \) and \( L_R \). Note that in the following discussion, the tree traversal will be pruned at \( n^L(v) \) or \( n^R(v) \) if they are null.

(a) If \( b_{uv} \) and \( b_{uw} \) do not cross \( l_v \) then we have \( v \in H_V \) (Theorem 4). The tree traversal will be continue with the subtree rooted at \( n^L(v) \), \( L_L \) remained as its left stack and \( v \) as the bottom element in its right stack. Similarly, the traversal of the subtree rooted at \( n^R(v) \) will have \( L_R \) remained as its right stack and \( v \) as the bottom element in its left stack.

(b) If \( b_{uv} \) and \( b_{uw} \) intersect each other above the bottom edge, then \( v \) is in \( H_B \), otherwise, if at least one of them crosses \( l_v \), \( v \) is a potential vertex (Theorem 4).

- If \( b_{uv} \) intersects \( l_v \), then \( n^L(v) \) is pruned (Theorem 5) and the \( NH \) represented by the subtree rooted at \( n^L(v) \) will be solved recursively if \( n^L(v) \) is not null. If \( b_{uv} \) does not intersect \( l_v \), the tree traversal on \( n^L(v) \) will be continued with \( v \) pushed into the right stack, \( L_R \).

- If \( b_{uw} \) intersects \( l_v \), then \( n^R(v) \) is pruned (Theorem 5) and the \( NH \) represented by the subtree rooted at \( n^R(v) \) will be solved recursively if \( n^R(v) \) is not null. If \( b_{uw} \) does not intersect \( l_v \), the tree traversal on \( n^R(v) \) will be continued with \( v \) pushed into the left stack, \( L_L \).

(c) If both \( n^L(v) \) and \( n^R(v) \) are pruned, all the vertices in \( L_L \) and \( L_R \) will become vertices in \( H_V \).

### 3.4 Complexity analysis

Our method for constructing the Constrained Voronoi diagram of a simple polygon \( P \) mainly relies on the efficiency of the identification of the \( INHs \) from an \( NH \). Since the identification for different \( INHs \) is executed recursively, we shall only consider the root \( INH \) of an \( NH \).

As described previously, when we traverse tree \( T_P \) of an \( NH \) to identify an \( INH \), we visit each
vertex of the \( INH \) exactly once. Those vertices have not been visited in the traversal of \( T_P \) cannot belong to the root \( INH \). Therefore, we only need to show that each visited vertex is tested in constant times in order to classify it as a vertex in \( H_V \) or in \( H_B \).

Let us consider a vertex \( v \). In the test, \( v \) can be classified into one of the three types: (i) \( v \in H_V \), (ii) \( v \in H_B \), and (iii) \( v \) is a potential vertex. In type (i), \( v \) is stored in the list of vertices representing \( H_V \). In type (ii), \( v \) is stored in the list of vertices corresponding to a particular \( H_B \). Note that each vertex in \( H_V \) or potential vertex can have a list of vertices corresponding to its associated \( H_B \). If the potential vertex, separating the two lists of vertices corresponding to two \( H_B \)'s, has been determined to be in \( H_B \), then these two lists of vertices have to be merged together. Vertex \( v \) will never be tested again. In type (iii), \( v \) is stored in the left or right stack and could be repeatedly tested when the descendants of \( v \) are visited (Action A). However, once vertex \( v \) is identified to be a vertex in \( H_V \) or \( H_B \), \( v \) will never be tested again. Thus, we can argue that the time for visiting a vertex is constant when amortized over a sequence of tests. To see this, our analysis assumes that one unit credit should have been assigned to each potential vertex in \( L_L \) and \( L_R \). Two unit credits are needed for each test, one for the cost in carrying the test itself and the other is for assigning to the vertex should it be identified as a potential vertex. The test on a vertex in \( L_L \) or \( L_R \) to determine whether or not it is in \( H_B \) (Action A) will be paid by the unit credit associated with the vertex.

It is not difficult to see that linked lists can be used to keep track the vertices in \( H_V \) and \( H_B \)'s. In particular, insertion and concatenation operations on \( H_V \) and \( H_B \)'s can be executed in constant time. The time complexity analysis for constructing of Voronoi diagrams of \( INH \), \( NH \), and \( P \) is obvious as described in the previous sections. We shall conclude the above analysis by the following theorem.

**Theorem 6** \( CDT(P) \) can be found in \( \Theta(|P|) \) time for simple polygon \( P \).
4 Concluding Remarks

In this paper, we presented a deterministic algorithm for finding the Constrained Delaunay triangulation of a simple polygon with \( n \) sides in \( O(n) \) time in the worst case. This may be the first linear-time algorithm for non-arbitrary triangulation of a simple polygon.

In the definition of Delaunay triangulation, we can check whether a triangulation is Delaunay by studying vertices within local proximity. It should not be surprising that the Delaunay triangulation and the constrained Voronoi diagram of a simple polygon can be done in linear time after given the Chazelle's horizontal visibility map, which links vertices within proximity together. The horizontal visibility maps are helpful to decompose the polygon into components such that 'divide and conquer' approach can be applied. However, if the decomposition of the polygon into components is not carefully done, interaction of the Voronoi diagrams of the components may be more than linear (even quadratic time). From Theorem 1, the partition of the polygon into components by chords has the advantage that the Voronoi diagrams of the components at the same level would not interact each other, i.e., horizontal interaction can be reduced. Moreover, because of the property of \( H_V \), interaction of Voronoi diagrams of components at different levels can also be confined, i.e., vertical interaction can be eliminated.

With our linear-time algorithm, the following related problems can also be solved efficiently.

(1) All nearest (mutual visible) neighbors of the vertices of a simple polygon.

(2) A shortest diagonal of a simple polygon.

(3) A largest inscribing circle of vertices of a simple polygon.

(4) The nearest vertex from a query point.

(5) Finding \( DT(S) \) if the Euclidean Minimum Spanning tree for a point set \( S \) is given.

(6) Finding standard Voronoi diagram for \( S' \) if the Voronoi diagram of a point set \( S \) is known, where \( S' \subset S \).
Acknowledgement: The authors would like to thank Bethany Chan and Siu-Wing Cheng for their patience in reading the first draft of this paper and their comments in improving the readability of the paper.

5 References

Appendix: The algorithm

The following global algorithm summarizes the above ideas.

Algorithm \textbf{Find\textit{(CDT}(P))}

\textbf{Input:} A simple polygon, denoted by \textit{P}.
\textbf{Output:} \textit{CDT}(P).
\textbf{Method:}
(1) Construct a tree \textit{T} of pseudo-normal histograms from \textit{P}.
   (* The decomposition is based on the horizontal and vertical visibility map of \textit{P}
   which can be obtained in [Chaz90]. Each \textit{PNH} is converted into an \textit{NH}. *)
(2) For (every \textit{NH} in \textit{T}, say \textit{H}) Do
   (a) Obtain a tree \textit{T}_\textit{P}.
      (* By using the horizontal visibility map of \textit{NH}. *)
   (b) Identify the \textit{INH} w.r.t. the bottom edge \textit{n}(\textit{v}_0) of \textit{NH} by \textbf{INH}\textit{(TP, n(v0))};
      (* The procedure \textbf{INH} traverses \textit{T}_\textit{P} from root \textit{n}(0) to produce an \textit{HV} and
      a set of attached \textit{HB}’s. The non-traversed part of \textit{T}_\textit{P} consists of a set of
      subtrees representing the smaller \textit{NH}’s of \textit{H} that contain the \textit{INH}’s of high levels. *)
   (c) Recursively apply procedure \textbf{INH} to each remaining subtree.
      (* The recursion results in a tree \textit{T}_\textit{I} of \textit{INH}’s, where each \textit{INH} consists of an \textit{HV} and
      a set of attached \textit{HB}’s. *)
      (* Obtain \textit{V}_\textit{c}(\textit{HV}) and \textit{V}_\textit{c}(\textit{HB}) [AGSS89, DjLi89]. Then, for all \textit{NH}’s obtain \textit{V}_\textit{c}(\textit{INH})
      by merging \textit{V}_\textit{c}(\textit{HV}) and its \textit{V}_\textit{c}(\textit{HB})’s. Merge all \textit{V}_\textit{c}(\textit{H}) to get \textit{V}_\textit{c}(\textit{P}) *)

In the procedure \textbf{INH}(\textit{TP, n(v0)}), the normal histogram \textit{H} is represented by tree \textit{T}_\textit{P} with root
\textit{n}() (the bottom edge). The resulting \textit{HV} is stored in a linked-list, denoted by \textit{C}, each resulting \textit{HB}
is stored in a linked list, denoted by \textit{Q}_z. For each potential vertex \textit{z} in \textit{LL} (\textit{LR}), \textit{Q}_x stores all those
vertices in \textit{HB} between \textit{z} and its next vertex in \textit{LL} (\textit{LR}). We use the notation \textit{Q}_x \leftarrow \textit{v} to represent
the action of inserting \textit{v} to the linked list \textit{Q}_x. When a potential vertex \textit{z} in \textit{LL(LR)} is identified as in
\textit{HB}, the two adjacent linked lists, \textit{Q}_x and \textit{Q}_y, will be concatenated together, where \textit{y} is the preceding
vertex of \textit{z} in \textit{LL} (\textit{LR}). Procedure \textbf{Action A} is to process \textit{LL} and \textit{LR} as described in page 19, the
vertices identified are put into its corresponding \textit{Q}_x.

\textit{u} \leftarrow \textit{v}_1; \textit{w} \leftarrow \textit{v}_n; \textit{v}_0 \leftarrow \textit{v}_1; \textit{C} \leftarrow (\textit{v}_1, \textit{v}_n); \textit{ST} \leftarrow (\textit{LL} \leftarrow \textit{v}_1, \textit{LR} \leftarrow \textit{v}_n); (* Initialization *)

\textbf{Procedure INH}(\textit{T, n(v0)}) (* return (C, Q) *)

\textbf{1 Action A (v, LL, LR, u_n, w_n)}
case (a) \((b_{uv} \text{ crosses } b_{vw} \text{ above } n(v_1))\) Then \(Q_v \leftarrow v;\)
\(\text{If (both } b_{uv} \text{ and } b_{vw} \text{ cross } l_v) \text{ Then Exit.}\)
\(*\) This branch contains no vertex of the \(INH\) *

2 Action B

\text{case (b) (both } b_{uv} \text{ and } b_{vw} \text{ do not cross } l_v) \text{ Then } C \leftarrow v.\)
\(*\) \(v\) belongs to the vertices of \(H_V.\) *

\text{case (c) } (b_{uv} \text{ or } b_{vw} \text{ crosses } l_v) \text{ Then } \text{Push}(v, ST);\)
\(*\) \(v\) is put on the potential vertex stack \(L_L\) or \(L_R.\) *

3 \(\text{If } (b_{uw} \text{ does not cross } l_v \text{ and } \overline{vw} \text{ is a diagonal}) \text{ Then}\)
\(\text{INH}(n^R(v), n(v_o));\)

4 \(\text{If } (b_{uv} \text{ does not cross } l_v \text{ and } \overline{uw} \text{ is a diagonal}) \text{ Then}\)
\(\text{INH}(n^L(v), n(v_o));\)
\(*\) further examine the subtrees of \(n(v)\) according to Theorem 4 *

EndProcedure
P 516.002854 C53
Chin, Yo-lun, Francis
Finding the constrained
delaunay triangulation and
constrained voronoi diagram of
a simple polygon in linear
time